جزوات دانشگاه صنعتی اصفهان

@jove_iut

www.jozve-iut.ir

Subject:
Date

توابع (Function in C):

• بخش هایی از برنامه را به صورت زیربرنامه هایی به شکل تابع می نویسیم.

فواید:
• پرهیز از نوشتن مکرر بخش هایی که چندین بار استفاده می شوند
• Team work ← شکستن برنامه به زیر بخش های جدا که هرکدام جداگانه قابل پیاده سازی باشند
• Debug کردن آسان تری شود

Functions in C:
- Library Functions توسط کتابخانه تعریف می شوند
- User-defined Functions توسط کاربر تعریف می شوند

@jozve_iut

■
- اعلان و معرفی تابع — Declaration and prototyping ← لیستی از آرگومان های تابع
- تعریف تابع
- فراخوانی تابع

< جنس خروج تابع >
int
float
void

مثال: ; (int mypow (int , int ← اعلان تابع (معرفی)

در خط فوق تابعی به نام mypow معرفی شده است که دو ورودی از جنس int می گیرد و خروجی از جنس int برمی گرداند.

تذکر: قبل از فراخوان تابع، تابع باید اعلان شده باشد. اگر تابع قبل از تابع main() تعریف شده باشد نیازی به اعلان تابع نیست

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

مثال: تابع توان :

```c
int mypow(int a, int b){
int p=1;
while(b>0){
p*=a; // p=p*a;
b--;
}
return p;
}
```

تعریف تابع

return p; ← کنترل را برمی‌گرداند بلافاصله بعد از دستور فراخوانی

return p; ← همراه با این برگشتن مقدار p را به تابع منتقل می‌کند

```c
#include<stdio.h>
int main(){
int a,b,c;
scanf("%d%d", &a, &b);
c=mypow(a,b);
printf("%d to the power of %d is %d", a,b,c);
return 0;
```

c=mypow(a,b); ← فراخوانی تابع

@jozve_iut

چند نکته :

۱. هر برنامه‌ای حداقل ۱ تابع دارد. (تابع main() ...)

۲. هر تابع را می‌توان در دل تابع دیگر فراخوانی کرد.

۳. محدودیتی برای فراخوانی یک تابع نیست

۴. ترتیب فراخوانی توابع و تعریف آن‌ها لزوماً یکسان نیست (بدیهی است)

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

چند نکته درباره‌ی return :

* یک تابع می‌تواند بیش از یک return داشته باشد

* لزوماً return در آخر تابع نیست.

مثال :

```
int myBow(int x){
    if(x%2)
      return (x+1)/2 ;
    else
      return (x)/2 ;
```

تذکر : در هر return فقط ۱ مقدار می‌تواند برگرداند

مثلاً : return(a,b,c) غلط است

@jozve_iut

مثال : تابعی بنویسید که عدد صحیح n را بگیرد و n! را برگرداند.

```
int myFact(int n){
 int f=1;
 while(n>0){
  f*=n; // f=f*n;
  n--;
 }
 return f;
}
int main(){
```


کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

تذکر:

```c
#include <stdio.h>
```

[اعلان توابع] → `#include <myheader.h>`

```c
int main(){

}
fun(1)
fun(2)
;
```

تذکر: تابع را می‌توان در دل خود نیز فراخوانی کرد.

$n! = n \times (n-1)!$ ⟸ امکان نوشتن برخی از توابع به فرم Recursive.

رابطه Recursive (بازگشتی)

مثال:

```c
int myrefact(int n)
int f;
if(n==1)
  f=1 ; //return 1;
else
  f=n * myrefact(n-1);
return f;
}
```

@jozve_iut

تمرین: تابعی بنویسید که عدد صحیح n را بگیرد و مجموع ارقام آن را برگرداند. به ۲ روش (مستقیم / بازگشتی)

بازگشتی → $f(n) = n\%10 + f(n/10)$

تذکر: توابع با جنس void مقداری را برنمی‌گردانند.

مثال:

```c
void message(){
  printf("Hello\n");
  printf("Enter your id:");
}
```

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

یادآوری: آدرس ها به بایت های حافظه تعلق می گیرند (عدد صحیح بدون علامت)

فراخوانی تابع
- call by value ⟸ مثال / با مقدار
- call by Reference ⟵ به جای مقادیر آرگومان ها می توانیم آدرس آن ها را به تابع منتقل کنیم / از طریق ارجاع
  - How? → Pointer / اشاره گر
  - چه فایده ای دارد؟ (به زودی)

مثال:

```
int i=3
printf("i=%d\n", i);
printf("Address of i is %u", &i);
```

%u → unsigned (int بدون علامت)

```
i=3
Address of i is 65520
```

(مثلاً)

مثال:

```
int *j; //Declaring a pointer to an int
int i=3;
j=&i;
```

j یک اشاره گر است. j در این مثال یک اشاره گر به عدد صحیح است (یک متغیر معمولی نیست)

Declare?
- int *p
- float *p
- char *p

← جنس متغیری که p به آن تعلق دارد.

مثال:

```
int i=3
int *j;
j=&i;
printf("%d",i);
printf("%d",*(&i));
printf("%d",*j);
```

معادل هم

PAPCO

@jozve_iut
کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

مثال:

```
void swap(int, int);
int main(){
int a=3 , b=2;
swap(a,b);
printf("\n a=%d\n b=%d", a, b);
return 0;
}
void swap( int a, int b){
 int c=a;
 a=b;
 b=c;
 printf("a=%d\t b=%d", a, b);
}
```

```
a=2   b=3
a=3
b=2
```

@jozve_iut

مثال:

```
void swapr(int* , int*);
int main(){
int a=3, b=2;
Swapr (&a, &b);
printf("a=%d\n b=%d", a, b);
void swapr( int *p , int *q ){      //p : address of a
int c=*p;                           // q : address of b
*p=*q;
*q=c;
}
```

```
a=3
b=2
```

نتیجه: فایده ی نشانگر: علاوه بر محدودیت Return و call by value.

Return فقط 1 مقدار را می‌توانست به تابع برگرداند.

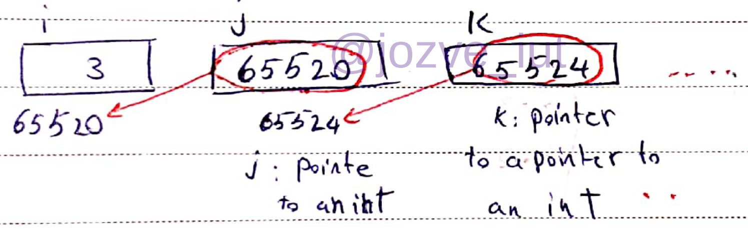
@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

```c
void areaperi(float, float*, float*);
int main(){
float r, A, P;
printf("Enter the radius of the circle:");
scanf("%d", &r);
areaperi(r, &A, &P);
printf("\n Area=%f\n perimeter=%f", A, P);
return 0;
}
void areaperi(float r, float *a, folat *p){
*a= 3.14*r*r;
*p=2*3.14*r;
}
```

شکل ۲

Area= ____
Perimeter= ____




```c
int i=3;
int *j, **k;
j=&i;
k=&j;
printf("%d", i);
printf("%d", *j);
printf("%d", **k);
```

معادل

PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

Preprocessing in C (پیش پردازش):

`#include < .h>`

• دستورهایی که با # شروع می شوند.

`#define Macro` ← Macro Expansion (Macro)

قالب

اسم ماکرو

• قبل از اجرای برنامه اصلی توسط کامپایلر اجرا می شوند

• در پایان ; ندارند

•

مثال:

```c
#include<stdio.h>
#define UPPER 5
int main(){
int i;
for(i=0; i<=UPPER; i++)
  printf("%d\n",i);
return 0; }
```

0
1
2
⋮
5

★ کامپایلر در وقت پیش پردازش، قبل از اجرای برنامه اصلی اگر با تعریف یک ماکرو مواجه شود، در متن برنامه هر جا که قالب ماکرو (اسم ماکرو) باشد معادل آن را جایگزین می کند.

تذکر، معمولاً برای اسم ماکرو از حروف تمام بزرگ لاتین استفاده می شود زیرا در متن برنامه اسم ماکرو بهتر به چشم بیاید.

@jozve_iut

کاربرد Macro: ① برنامه تمیزتر و مرتب تر می شود

② سهولت برخی پارامترهای ثابت در برنامه (مثلاً دقت یک عدد اعشاری)

③ (فانتزی) برحسب سلیقه ی برنامه نویس (تذکر: عملگرها را نیز می توان به صورت ماکرو در نظر گرفت)
#define AND &&

تفاوت تعریف کردن متغیر و define؟

① اساساً به لحاظ فلسفی مشکل دارد

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

(۴) سرعت کار کردن با ثابت ها (define) بیشتر است.

(۵) به هر حال مقدار متغیر می تواند به صورت ناخواسته در طول برنامه تغییر کند

مثال:
```
#define WARNING printf("Be careful").
#define AND &&
#define SHART (x<5 AND x>1)

int x;
scanf("%d", &x);
if (SHART)
    WARNING;
```
(که باید مثل هم باشن)

۱. حتی برخی دستور ها را می توان به صورت ماکرو در نظر گرفت (printf ....)

۲. در قسمت بسط یک ماکرو می توان از ماکروی دیگری که قبلاً تعریف شده استفاده کرد.

۳. حتی یک شرط را می توانیم به صورت ماکرو تعریف کنیم. (*)

@jozve_iut

Multi-line Macros:
```
#define HLINE for(int i=0; i<=40; i++)\
        Print("-");
```
کشیدن یک خط افقی با ۴۰ کاراکتر

Macros:
- Simple Macros (همین هایی که تا الان دیدیم)
- Macros with Argament (ماکروهای دارای آرگومان)

مثال:
```
#define AREA(x) (3.14*x*x)
int main(){
  float r, A;
  scanf("%f", &r);
  A= AREA(r);
  printf("%f", A);
  return 0;
}
```
چون پرانتز دارد یعنی آرگومان دارد

ضرورت پرانتز ها: برای جلوگیری از ابهام در تقدم اولویت عملگرها بسط ماکروی دارای آرگومان را در پرانتز قرار می دهیم.

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

سؤال: Macros with Argument v.s. Functions ?

Macros with Argument	Functions
سرعت بالاتر	انعطاف پذیرتر
محدود	جامع
حافظه ی بیشتری مصرف می کند	حافظه ی کمتری مصرف می کند

آرایه ها: Arrays in C:

- مجموعه ای از المان های مرتبط به (هم نوع)، مثلاً مجموعه ای از اعداد صحیح

- اعلان آرایه: جنس آرایه اسم آرایه [اندازه آرایه]

مثال:
```
int num[4]; //num[0], num[1], num[2], num[3]
```

مقداردهی اولیه ی آرایه: مثال:
```
int num[4] = {1,3,0,-1}
```

@jozve_iut

تذکر: اگر همزمان با اعلان آرایه مقداردهی اولیه آن نیز انجام شود، نوشتن اندازه آرایه داخل براکت الزامی نیست.

```
int num[4] = {1,3,0,-1}
int num[ ] = {1,3,0,-1}
```
(الزامی)

مثال:
```
int num[30], i, sum;
  printf("Enter your numbers: ");
 for(i=0; i<=29; i++){
    scanf("%d", &num[i]);}
printf("The numbers you entered are: ");
for(i=0; i<=29; i++){
  printf("\n%d", num[i]);
  sum = sum + num[i]; }
av g = sum/30
```

P4PCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

تذکر: اندیس آرایه از صفر تا d-1 (d اندازه آرایه) در تعریف ذیل که خود باید اعلان شده باشد. اندیس از 0 شروع می‌شود

تذکر (مهم): اطمینان از اینکه اندیس آرایه از محدوده مجاز آن بیرون نرود به عهده برنامه‌نویس است

(کامپایلر کنترلی روی آن ندارد و پیغام خطا نمی‌دهد)

مثال:
```
float num[10],i;
for(i=0; i<30; i++){
 scanf("%f", &num[i]);
```

ارسال المان های آرایه به تابع ← المان به المان به صورت یکجا

مثال:
```
void disp(int);
int num[10],i;
for(i=0; i<=9; i++){
 scanf("%d", &num[i]); }
for(i=0; i<=9; i++){
 disp(num[i]); //by value
}
void disp(int n){
    printf("\n%d",n); }
```

مثال:
```
void disp(int *)
 :
disp(&num[i]); //by Refrence
```

```
void disp(int *p){
    print("\n %d", *p);
}
```

PAPCO

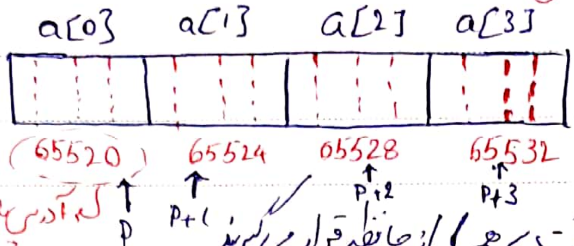
@jozve_iut

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

نحوه قرار گرفتن المان‌های آرایه در خانه‌های حافظه:

(مثلاً کامپایلر 32 بیتی)

`int a[4];`



Base Address: که آدرس پایه آرایه &a[0]

مهم: المان‌های آرایه در خانه‌های متوالی (پشت سرهم) از حافظه قرار می‌گیرند

برخی عملیات مجاز بر روی پوینترها:

• اضافه کردن عدد ثابت به پوینتر: اگر به پوینتر 1 واحد اضافه شود به اولین خانه همجنس بعد از خودش اشاره خواهد کرد

تحویل دادن آرایه به تابع، مثال:

```
int a[4], i;
for (i=0; i<=3; i++) {
  scanf("%d", &a[i]); }
dispall(&a[0], 4);

void dispall(int *p, int d) {
 int i;
 for (i=0; i<=d-1; i++)
   printf("%d\n", *(p+i)); // printf("%d\n", *p)
                            // p++
 }
}
```

تذکر: در زبان C اسم آرایه (یا بعدی) همان آدرس پایه آرایه را می‌دهد:

a[i] معادل *(&a[i])
معادل *(&a[0]+i)
معادل *(a+i) ← آدرس پایه

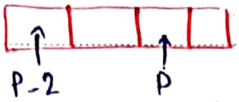
کامپایلر خودش بصورت پوینتری به a[i] دسترسی پیدا می‌کند

PAPCO
@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

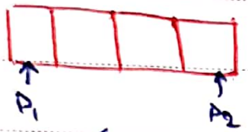
تذکر: المان های یک آرایه تا وقتی که مقداردهی نشده باشند (قاعدتاً) حاوی Garbage value هستند (صفر نیست)

برخی عملیات مجاز دیگر بر روی اشاره گرها:

* تفریق عدد ثابت از اشاره گر



* تفریق دو اشاره گر از هم: به شرطی مجاز است که هر دو اشاره گر به خانه‌هایی از یک آرایه مشترک اشاره کنند.



P2 - P1 : 3 به عنوان فاصله بین دو المان

* مقایسه‌ی دو اشاره گر: وقتی مجاز است که به خانه‌های همجنس اشاره کنند مثلاً خانه‌های یک آرایه

مثال:

```
int a[5];
int *P1, P2;
P1 = &a[2];
P2 = a+2; // a ≡ &a[0]
if(P1==P2)
  printf("Both pointers point to the same place");
```

Both pointers point to the same place

آرایه دو بعدی:

جنس آرایه اسم [تعداد سطر] [تعداد ستون]

مثال: int a[4][3];

a[0][0]	a01	a02
a10	a11	a12
a20	a21	a22
a30	a31	a32

تذکر: در آرایه دو بعدی اندیس سطر و ستون از صفر شده است

PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject
Date

مقداردهی اولیه آرایه دو بعدی:

مثال:
```
int num[4][2]={ {16,1},
                {25,3},
                {17,2},
                {19,0} };
```

الزامی

به طور معادل: `int num[4][2]={16,1,25,3,17,2,19,0};`

اختیاری

تذکر: در آرایه دو بعدی، اگر مقداردهی اولیه همزمان با اعلان آرایه انجام شود، نوشتن تعداد ستون ها الزامی و تعداد سطرها اختیاری است.

خواندن المان های آرایه ۲ بعدی از ورودی:

ورودی:
```
int num[10][2],i,j;
for(i=0;i<=9;i++){
  for(j=0;j<=1;j++){
    scanf("%d",&num[i][j]);
  }
}
```

← اگر تعداد ستون زیاد باشد می توان کل المان های یک سطر را در یک مرحله با هم گرفت (در جای حلقه)

@jozve_iut

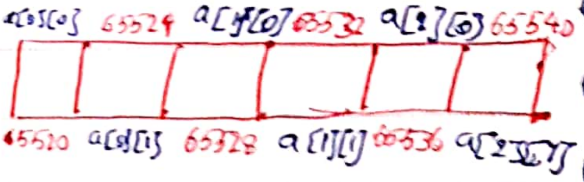
خروجی:
```
for(i=0;i<=9;i++){
  for(j=0;j<=1;j++){
    printf("%d",num[i][j]);
  }
  printf("\n");
}
```

تذکر مهم: المان های آرایه دو بعدی نیز در خانه های متوالی از حافظه قرار دارند.

مثال: `int a[3][2];`

a[0][0]	a[0][1]
a[1][0]	a[1][1]
a[2][0]	a[2][1]

یا مثلاً:

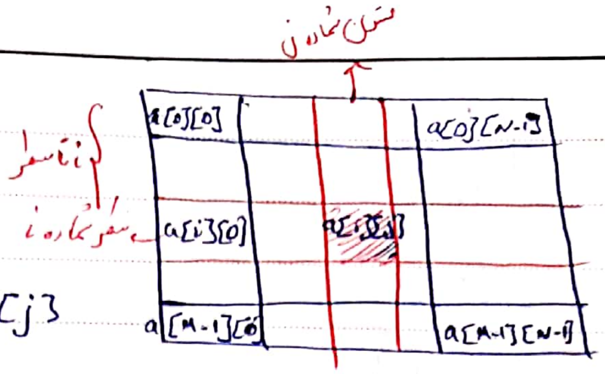


PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject
Date

مثال: int a[M][N];



*(&a[0][0]+i*N+j) ≡ a[i][j]

آدرس پایه سطر i ام &a[i][0]

a[i]

نکته: آرایه دو بعدی را می توانیم به صورت آرایه ای از آرایه های 1 بعدی در نظر بگیریم. هر سطر یک آرایه.

معادل ها: a[i][j]

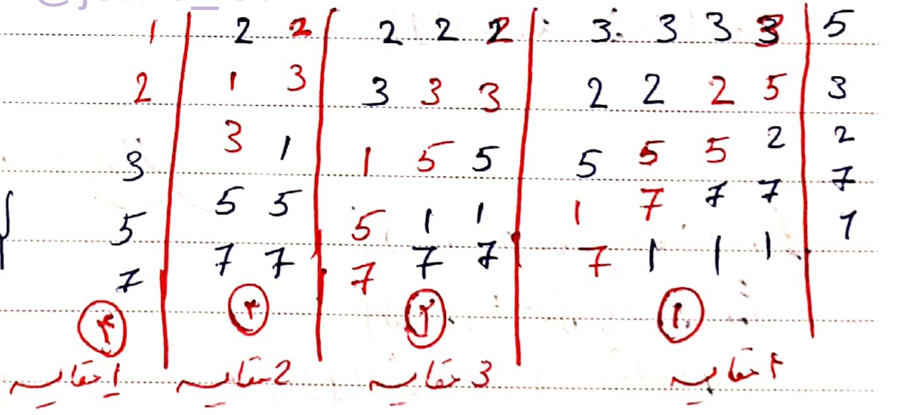
*(&a[0][0]+i*N+j)

*(a[i]+j)

((a+i)+j)

مثال: Sort کردن یک آرایه n تایی از اعداد:

```
#define N 100
int a[N], i, j, temp;
for (i=0; i<=N-1; i++){
  scanf("%d", &a[i]); }
for(i=1; i<N; i++){
  for (j=0; j<N-i; j++){
     if(a[j]>a[j+1]){
  temp = a[j];
  a[j] = a[j+1];
  a[j-1]= temp;
     }
  }
}
```



آخرین مقایسه در هر مرحله: a[N-i-1] در N-i مقایسه

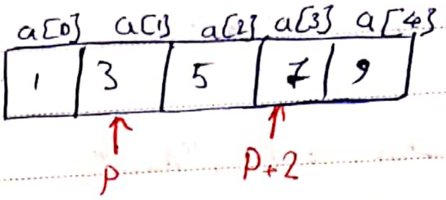
PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

مثال:
```c
int a[5] = {1, 3, 5, 7};
int *p = &a[1];   // int *p; p = &a[1];
printf("%d", p[2]);
```
p[2] → *(p+2)

7



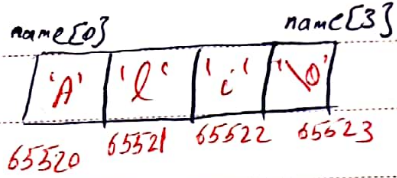
strings (رشته‌ها):

رشته: آرایه ای از کاراکترها که به کاراکتر Null ('\0') ختم می شوند.

مثال:
```c
char name[] = {'A', 'l', 'i', '\0'};
```
'\0' ← پایان رشته

با توجه به اینکه رشته خود یک آرایه است، در مقداردهی هم زمان با اعلام رشته، ذکر اندازه ی آرایه اختیاری است.

کاراکتر	کد اسکی
'\0'	0



تذکر: عناصر رشته در خانه های متوالی از حافظه.

توجه: با توجه به کاربرد زیاد رشته ها در زبان c یک مبانی برای اعلان و مقداردهی هم زمان رشته در نظر گرفته شده.

```c
char name[] = "Ali";
```

در روش فوق نیازی به نوشتن Null نیست. در واقع خود کامپایلر Null را اتوماتیک در پایان رشته قرار می دهد (وقتی "" را می بندیم)

مثال:
```c
char name[] = "Ali";
int i;
for(i=0; i<=3; i++){
    printf("%d", name[i]); }
```

بازنویسی برنامه بالا مستقل از سایز آرایه:
```c
int i=0;
while(name[i] != '\0'){
    printf("%c", name[i]);
    i++;
}
```

PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

تذکر: اسم رشته آدرس پایه رشته را می دهد (رشته یک آرایه است) name[i] معادل *(name+i)

مثال: char str[] = "Amir";
printf("%s", str); ← آدرس پایه رشته // معادل printf("%s", &str[0]); → Amir

printf("%s", &str[1]); → mir

چاپ ثابت رشته ای با printf:

مثال: printf("Hello");

با استفاده از %s: printf("%s", "Hello");

گرفتن رشته از ورودی با تابع scanf:

مثال: char yourname[15];
printf("%s", "Enter your name: ");
scanf("%s", yourname); ← آدرس پایه رشته
printf("\n Hello %s", yourname);

Enter your name : Ali
Hello Ali

تذکر ۱: مسئولیت اینکه رشته دریافتی در کل مورد نظر جا شود با برنامه نویس است

تذکر ۲: رشته های دو بخشی را نمی توان با تابع scanf دریافت کرد. یک راه حل استفاده از تابع gets

char name[15];
gets(name);

C programming
↓ Space — ۲ بخشی

ذخیره رشته به کمک Pointer:

char *p = "Ali";
//char *p;
p = "Ali";

| 'A' | 'l' | 'i' | '\0' |
p↑

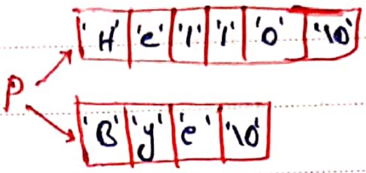
PAPCO
@jozve_iut
@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

۲- تفاوت این دو روش :

```c
char str1[] = "Hello";
char str2[10];
E ✗ str2 = str1;
E ✗ str1 = "Bye";
```

```c
char *p = "Hello";
char *q;
q = p;
p = "Bye";
```



برخی توابع برای رشته ها:

```c
<string.h>
<stdlib.h>
```

تابع ۱: strlen (آدرس پایه رشته): این تابع آدرس پایه رشته را گرفته و طول رشته (تعداد کاراکترهای آن بدون احتساب Null) را برمی گرداند.

مثال:

```c
char name[] = "Ali";
int l = strlen(name);
printf("%d", l);
```

3

نسخه بومی strlen :

```c
int mystrlen(char *p){
    while(*p != '\0'){
        l++;
        p++;
    }
    return l;
}
```

تابع ۲: strcpy : strcpy(target, source) :

(target: آدرس پایه رشته مقصد — source: آدرس پایه رشته مبدأ)

این تابع آدرس پایه رشته مبدأ و مقصد را گرفته و رشته مبدأ را در رشته مقصد کپی می کند

تذکر: اندازه رشته مقصد باید به قدر کافی بزرگ باشد (مسئولیت برنامه نویس)

PAPCO

@jozve_iut
@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

مثال:
```
char name1[] = "Ali";
char name2[10];
strcpy(name2, name1);
printf("%s", name2);
```
Ali

شبیه‌سازی strcpy:
```
void mystrcpy(char *t, char *s){
  while (*s != '\0'){
    *t = *s;
    t++;
    s++;
  }
  *t = '\0';
}
```

تابع ۳: strcat(target, source) (مبدا / مقصد)

مثال:
```
char str1[15] = "C-";
char str2[] = "programming";
strcat(str1, str2);
printf("%s", str1);
```
C-programming

شبیه‌سازی strcat (تمرین)

راهنمایی: شروع کپی کردن در رشته مقصد از Null ...

تابع ۴: strcmp (target, source): این تابع آدرس دو رشته را گرفته، اگر دو رشته یکسان باشند 0 را برمی‌گرداند؛ در غیر این صورت اختلاف کد اسکی (با علامت آن) اولین جایی که دو رشته با هم فرق دارند را برمی‌گرداند.

PAPCO

@jozve_iut
کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject:
Date

مثال:
```
char name1[] = "Amir";
char name2[] = "Amin";
int i,j,k;
i = strcmp(name1, "Amir");
j = strcmp(name1, name2);
k = strcmp(name2, name1);
printf("%d\n%d\n%d", i,j,k);
```

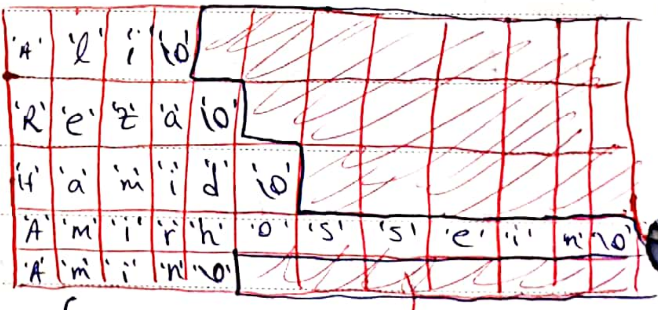
خروجی
0
4
-4

+4: n o p q r

کاربرد:
- برای تطبیق رشته ← password / جستجو در سیستم ها
- مرتب سازی الفبایی

مثال: برنامه ای بنویسید که اسم ورودی را با اسم های داده شده در برنامه چک کند

```
char name[][12] = {"Ali", "Reza", "Hamid", "Amirhossein", "Amin"}; //List of name
int flag=0, i;
printf("%s", "Enter your name:");
char yourname[15];
scanf("%s", yourname);
for (i=0; i<=4; i++){          & names[i][0]
  if (strcmp(yourname, name[i])==0){
    flag=1;
    printf("\n%s", "welcom to the program");
    break; }
}
if (!flag)
printf("\n%s", "sorry!. your name isn't in the list");
```



wastage (حافظه اضافه تلف شده)

Arrays of pointers to string: بهینه سازی حافظه با
← صفحه ی بعد

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

Subject :
Date

با پوینتر داری:

```
char *names[] = {"Ali",
                 "Reza",
                 "Hamid",
                 "Amirhossein",
                 "Amin"};
```

'A'	'l'	'i'	'\0'

65520 → names[0]

'R'	'e'	'z'	'a'	'\0'

65524 → names[1]

'H'	'a'	'm'	'i'	'd'	'\0'

names[2] . . . .

مثال: حال می‌خواهم در برنامه قبل جای اسم ها را (سطرها را) جابجا کنم. (اسم جای سطر دوم و پنجم)

```
char names[][12]={"Ali",
                  "Reza",
                  "Hamid",
                  "Amirhossein",
                  "Amin"};
int j;
char temp;
printf("Original:%s\t%s", names[1], names[4]);
for(j=0; j<=11; j++){
  temp = names[1][j];
  names[1][j] = names[4][j];
  names[4][j] = temp; }
printf("\n New:%s\t%s", names[1], names[4]);
```

} 12 exchange

Original : Reza Amin
New : Amin Reza

حال با استفاده از Pointer:

```
char *names[] = { "   "   " }
int j;
char *temp;
printf("                    ");
temp = names[1];
names[1] = names[4];
names[4] = temp;
```

} 1 exchange
4 pointer

PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

A list of strings

Two-Dimensional Array	Array of pointers to strings
دورریز حافظه	استفاده بهینه تر از حافظه
exchange به صورت کاراکتر به کاراکتر (کندتر)	exchange با دو تر رشته ها (سریعتر)
name[i][j]	name[i][j]
	✓ *(name[i]+j)
امکان خواندن سطرها با scanf	مشکل خواندن scanf

Structures (ساختارها):

مجموعه ای از المان های غیرهم جنس.

مثال: دریافت اطلاعات ۳ کتاب و ذخیره آن ها:

با معلومات فعلی: char names[3]; int pages[3]; float price[3];

```
char names[3];
int pages[3];
float price[3];
printf("%s","Enter the name, page and price of 3 books");
for(int i=0 ; i<=2 ; i++){
scanf("%c%d%f", &names[i], &pages[i], &price[i]);
```

با struct:

```
struct book{
  char names;
  int pages;
  float price;
};
struct book b1,b2,b3;
```

structure type declaration

book: متغیرهایی از جنس ساختار

structure variable Declaration

ادامه برنامه:

```
scanf("%c%d%f", &b1.name, &b1.pages, &b1.price);
```

همین ترتیب برای b2 و b3

Structures: نکته:

```
struct اسم ساختار {
  نوع المان  اسم المان ;
  نوع المان  اسم المان ;
};
struct اسم نوع ساختار  اسامی متغیرهای با آن ساختار
```

PAPCO

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

چند تذکر:

۱- ; پایان اعلان نوع ساختار (بعد از }) الزامی است

۲- اعلان نوع ساختار و اعلان متغیرهایی از آن نوع ساختار می‌توانند همزمان (باهم) صورت بگیرند

مثال:
```
struct book{
    char name;
    int pages;
    float price;
}b1,b2,b3;
```

۳- در حالت فوق نوشتن اسم نوع ساختار (book) در اینجا اختیاری است اما در این صورت دیگر امکان تعریف متغیرهایی از این نوع ساختار در جاهای پایین‌تر برنامه نیست.

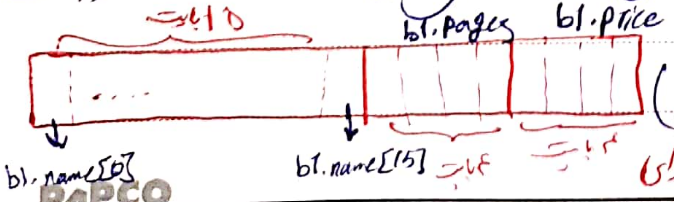
@jozve_iut

۴- معمولاً در برنامه‌های بزرگ، اعلان نوع ساختارها در آغاز برنامه قبل از تعریف متغیرها و توابع نوشته می‌شود بعضی از اوقات تمامی اعلان نوع ساختارهای مختلف در یک Header file نوشته می‌شود و در آغاز برنامه include می‌شود.

مثال: #include <mystructs.h>

چیدمان المان‌های یک متغیر از نوع ساختار در حافظه:

تذکر مهم: المان‌های یک متغیر از نوع یک ساختار مشخص در خانه‌های متوالی از حافظه قرار دارند (به همان ترتیبی که المان‌ها در نوع ساختار declare شده‌اند)



در مثال بالا برای کامپایلر ۳۲ بیتی

PAPCO

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

تذکر: صرف اعلان یک نوع ساختار حافظه ای به آن تخصیص داده نمی شود.

وقتی که متغیرهایی از جنس ساختار مشخص اعلان می شوند خانه های حافظه متناسب با نوع اعلان های ساختار به متغیرها تخصیص داده می شود

Array of structures (آرایه ای از ساختارها):

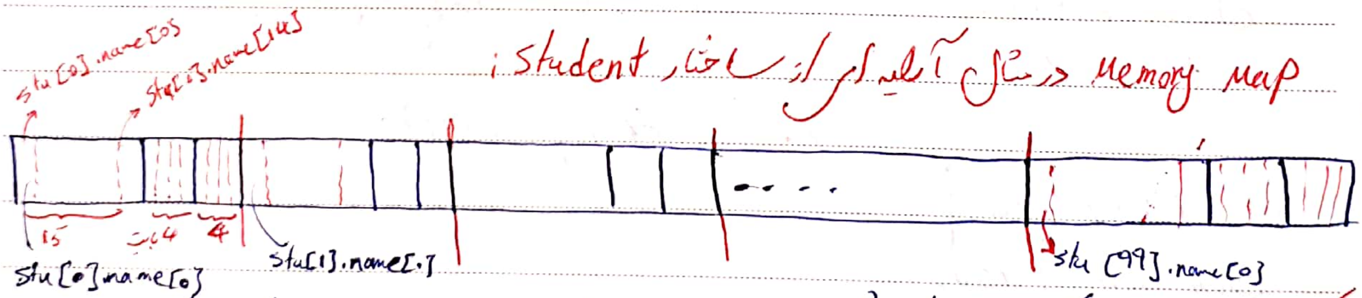
مثال:

```
struct student{
    char name[15];
    int unit;
    float avg;
};
struct student stu[100]; //stu[0], ... , stu[99]
for(int i=0 ; i<=99; i++){
scanf("%s%d%f", stu[i].name, &stu[i].unit, & stu[i].avg);
```

(۱۰۰ متغیر از نوع ساختار student)

@jozve_iut

در مثال آرایه ای از ساختار student: Memory map



تذکر: متغیرهای از نوع یک ساختار مشخص را می توان همراه با اعلان مقدار دهی اولیه نیز کرد

```
struct student{
char nam[15];
int unit;
float avg;
}
struct student stu1={"Ali",75,16.2};
```

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

مثال: struct employee{
```
    char name[15];
    int age;
    float salary;
};
struct employee emp1, emp2, emp3={"Reza", 37, 1325.6};
//Copy element_by_element
strcpy(em2.name, emp3.name);
emp2.age = emp3.age;
emp2.salary = emp3.salary;
```
emp1 = emp2;

تذکر: می توان یک متغیر از نوع ساختار مشخص را به متغیری از همان نوع ساختار تخصیص داد

Nested structure: ساختارهای تودرتو

ارجاعی ← void display(struct employee);

مثال:
```
struct address{
    char phone[12];
    char city[15];
    int pcode;
};
struct employee{
    char name[15];
    struct address a;
};
```

ادامه struct employee emp1={"Ali", "03139122210", "Isfahan", 84156};

printf("%s", emp1.a.phone);

۲ عملگر دات

فراخوانی تابع display(emp1);

@jozve_iut | کانال مرجع جزوات دانشگاه صنعتی اصفهان

Subject:
Date

تحویل یک متغیر ساختار به تابع:
- به صورت المان به المان به کمک پوینترها (از قبل بلد بودیم)
- به صورت یک جا

مثال:
```c
void display(struct employee e){
    printf("%s\t %s", e.name, e.a.phone);
}
```

PAPCO

@jozve_iut

کانال مرجع جزوات دانشگاه صنعتی اصفهان | @jozve_iut

جزوات دانشگاه صنعتی اصفهان

@jove_iut

www.jozve-iut.ir